Date / /
Page No.

Hopfield Nets :-

Discrete Hopfield Nets, this type of network was designed by J. J. Hopfield in 1982. The topology of Hopfield n/w is very simple. It has 'n' neurons which are all h/w worked with each other. A hopfield n/w is able to recognize unclear picture correctly. However only one picture can be stored at a time. In practical applications one must assume that many picture will be given, which have to be stored and then classified.

This net is a fully interconnected neural net with each unit connected to every other unit. The net has symmetric weight with no self connections i.e all the diagonal elements of weight matrix of Hopfield net are zero

$$w_{ij} = w_{ji} \text{ and}$$
$$w_{ii} = 0$$

The two main differences b/w Hopfield net and associative net (recurrent associative net) are that in the Hopfield net

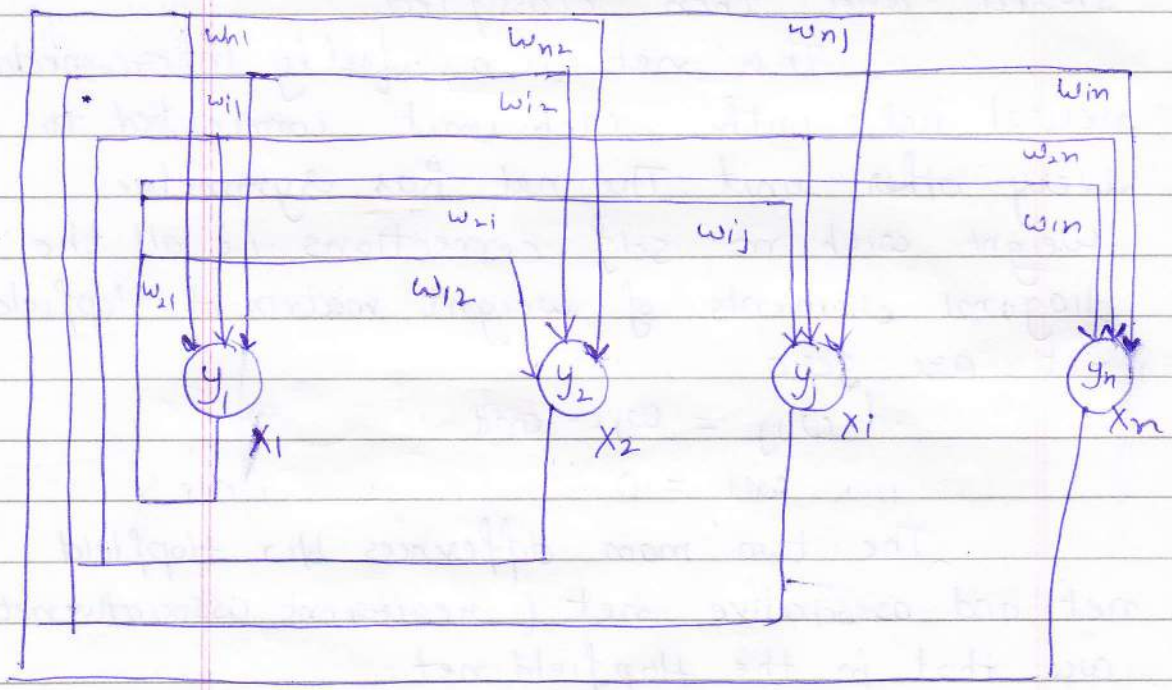
> only one unit updates it activation at a time and each unit continues to receive an external signal in addition to the signal from the other unit in the net.

Teacher's Signature

The asynchronous discrete time updating of the units allows a f^n known as an energy f^n or Lyapunov f^n to be found for the net this f^n proves that the net will converge to a stable set of activations.

Architecture of Discrete Hopfield Net :-

The architecture of the discrete Hopfield net is shown as :-



The architecture consist of 'n' no of x inputs neurons and Y o/p neurons. It should be noted that apart from receiving a signal from i/p. The Y_i

Teacher's Signature

neurons receive signal from o/p. The Y_i neurons also. This the form its other o/p neurons. Thus their same for all other o/p neurons. The their exists a feedback o/p being returned to each o/p neuron. That is why Hopfield network is called a feedback n/w

<u>Training Algorithm</u>:- Discrete Hopfield net is described for both binary as well as bipolar vector pattern. The weight matrix to store the set of binary o/p pattern

$S(P), P=1, \ldots, P$, where

$S(P) = (S_1(P), \ldots S_i(P) \ldots S_n(P_i))$

can be determined with the help of hebb rule. The weight matrix can determined by the formula

$$w_{ij} = \sum_{i0} (2S_i(P)-1)(2S_j(P)-1) \text{ for } i \neq j$$

and $w_{ii} = 0$

For bipolar o/p pattern, the weight matrix is given by,

$$w_{ij} = \sum_{P} S_i(P) S_j(P) \text{ for } i \neq j \text{ \& } w_{ii} = 0$$

Date / /
Page No.

Application Algorithm :- The weight to be used for the application algorithm are obtained from the training algorithm. Then the activations are set for the i/p vector. The net input is calculated and applying the activations. The o/p is calculated. The o/p is broadcasted to all other units. The process is repeated until the convergence of the net is obtained. The application algorithm of discrete Hopfield net is given as follows

Step 1: Initialize weight to store pattern while activations of net are not converged perform step 2 to 8

Step 2: for each i/p vector X, repeat steps 3 to 7

Step 3: Set initial activations of the net equal to external input vector X, $Y_i = X_i$ $(i = 1, \ldots n)$

Step 4 Perform steps 5 to 7 for each unit y_i

Step 5 Compute the net input

$$Y_{in_i} = X_i + \sum_j y_i - w_{ji}$$

Teacher's Signature

Step 6: Determine activation (o/p signal)

$$y_i = \begin{cases} 1 & \text{if } y_{in_i} > \theta_i \\ y_i & \text{if } y_{in_i} = \theta_i \\ 0 & \text{if } y_{in_i} < \theta_i \end{cases}$$

Step 7: Broadcast the value of y_i to all other units

Step 8: Test for convergence

The value of threshold θ_i is usually taken to be zero. The order of update the unit is random, but each unit must be updated at the same average rate.

Energy :- Hopfield has made an analogy b/w the training. The neural n/w and minimizing a value which has been interpreted as its global energy.

A neuron that has a large +ve weight associated with it is likely to fire when the input to the neuron is one i.e a large +ve weight means that the neuron fires easily with a stmulus on one of its inputs. On the other hand, an input with a

Teacher's Signature

small +ve value will probably not be able to make the neuron fire on its own. In this case a lot of stimulas or energy is required. Finally input with -ve weights will inhibit the neuron and so even more energy is required on the inputs to create fire

It is evident that energy in a neural net is related to the -ve sum of all the weighted inputs of the neurons. For a single neuron, the energy at a particular time is defined as

$$E = -y[x_2 w_2 + x_3 w_3 + \cdots + x_n w_n + \theta + x_i]$$

<u>Energy functions</u>:- The energy f^n for the Hopfield n/w shown as

$$E = \frac{-1}{2} \sum_{i \neq j} \sum y_i y_j w_{ij} - \sum x_i y_i + \sum_i \theta_i y_i$$

The change in energy is due to change in the state of the neuron. The neuron i in ΔE. This can be calculated using the following e^n

$$\Delta E = -\left\{ \sum_j y_i w_{ij} + x_i \theta_i \right] \Delta y_i$$

$$(net - \theta_i) \Delta y_i$$

Teacher's Signature

where Δy_i is the change in the o/p of the neuron i

To minimize E, we have to make sure that ΔE is always -ve

If the net value of the neuron i is greater that the threshold, The term in the brackets will be +ve) and making use of the activation fn. It can be further stated that the o/p of neuron i must change in +ve direction This means that Δy_i can be +ve or zero and ΔE must be -ve, hence the +ve energy must either decrease or stay constant

If the net value of the neuron is less than threshold. They Δy_i can be -ve or zero hence again. The energy must decrease or remain constant

Finally if the net equals its threshold θ_i, the energy remains constant.

Storage Capacity:- In the Hopfield net The no of binary input pattern That can be stored and recalled in a net with reasonable accuracy is given by

$P = 0.15n$ where n is the no. of neurons in the net

if bipolar pattern are used Then

$$P = \frac{n}{2\log_2 n}$$

<u>Example</u>:- Consider a vector (1 0 1 1) to be stored in a net. Test a discrete Hopfield net with mistakes in the first and fourth components (0 0 1 0) of the stored vector.

<u>Sol^n</u> Initialize weight to store patterns,

<u>Step 1</u> Convert binary input to bipolar & find weight

$$w = s^T s$$

$$w = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 1 & 1 \\ -1 & 0 & -1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 1 & 0 \end{bmatrix}$$

<u>Step 2</u> The input vector is $X = (0\ 0\ 1\ 0)$

<u>Step 3</u> $y = (0\ 0\ 1\ 0)$

<u>Step 4</u> Choose unit y_i to update its activation

Teacher's Signature

Charlie Date / / Page No.

step 5 $y_{-in1} = x_1 + \sum_j y_j w_{j1} = 0+1 = 1$

step 6: $y_{in1} > 0 \Rightarrow y_1 = 1$

step 7: $y = (1\ 0\ 1\ 0)$

step 4: choose unit y_4 to make update its activation

step 5: $y_{-in4} = x_4 + \sum_j y_j w_{j4} = 0 + 2 = 2$

step 6: $y_{-in4} > 0 \Rightarrow y_4 = 1$

step 7: $y = (1\ 0\ 1\ 1)$

step 4: choose unit y_2 to updat its activation

step 5 $y_{-in2} = x_2 + \sum_j y_j w_{j2} = 0 + (-1-1-1) = 0-3 = -3$

step 6 $y_{-in2} < 0 \Rightarrow y_2 = 0$

step 7: $y = (1\ 0\ 1\ 1)$

step 4: choose unit y_3 to updat its activation

step 5: $y_{-in3} = x_3 + \sum_j y_j w_{j3} = 1 + (2) = 3$

step 6: $y_{-in3} > 0 \Rightarrow y_3 = 1$

step 7: $y = (1\ 0\ 1\ 1)$

Teacher's Signature

step 8: Test for convergence

The further interaction can be carried out & the convergence can be checked.

<u>Example</u> - Design a Hopfield n/w for 4 bit bipolar pattern The training pattern are

I Sample $S_1 = [1 \quad 1 \quad -1 \quad -1]$

II Sample $S_2 = [-1 \quad 1 \quad -1 \quad 1]$

III Sample $S_3 = [-1 \quad 1 \quad -1 \quad 1]$

Find the weight Matrix & The energy f^n for these input samples. Determine the pattern to which the sample $S = [-1 \quad 1 \quad -1 \quad -1]$ associates

<u>Sol^n</u> -

$$X = \begin{bmatrix} 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

$$w = \frac{1}{4} \sum_{i=1} X_i X_i^T = X^T X$$

$$= \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 1 & 1 & -3 \\ 1 & 3 & -1 & -1 \\ 1 & -1 & 3 & -1 \\ -3 & -1 & -1 & 3 \end{bmatrix}$$

Teacher's Signature

calculating energy of the n/w for each pattern

$$E = \frac{-1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} x_i w_{ij} x_i t$$

Energy for sample pattern 1 is

$$S = [1 \quad 1 \quad -1 \quad -1]$$

$$E_1 = \frac{-1}{2}[18] = -9$$

Energy for pattern 2 is $S = [-1 \quad 1 \quad -1 \quad 1]$

$$E_2 = \frac{-1}{2}[22] = -11$$

Energy for pattern 3 is

$$S = [-1 \quad -1 \quad -1 \quad 1]$$

$$E_3 = \frac{-1}{2}[26] = 13$$

Now test vector is given by

$$S = [-1 \quad 1 \quad -1 \quad -1]$$

$$E = \frac{-1}{2}(10) = -5$$

$E = -5$ is less than -9 as expected

By asynchronous updation method

Net $\eta p = S \times w$

$$= [-1 \quad 1 \quad -1 \quad -1] [w]$$

$$[0 \quad 4 \quad -6 \quad 0]$$

Teacher's Signature

Date / /
Page No.

Apply Threshold limit $\theta = 0$
Therefore O/P = $[-1 \quad 1 \quad -1 \quad -1]$

Hence the pattern is closely related or associated with pattern 1

Stability :- As with other n/w's the weights b/w layers in the n/w may be considered to form a matrix w. Cohen and Grossberg (1983) have shown that recurrent n/w's are stable if the matrix is stable symmetrical with zero on its main diagonal that is if $w_{ij} = w_{ji}$ for i not equal to j and $w_{ii} = 0$ for all i

The stability of such a neural n/w may proven through an elegant mathematical with zero technique Suppose a f^n can be found that always decreases each time the n/w changes state Eventually the f^n must reach a minimum and stop, thereby ensuring That the n/w is stable The f^n that follows is called Lyapunov f^n & works in just such a manner on The recurrent n/w's presented above

Teacher's Signature

$$E = (-1/2) \sum_i \sum_j w_{ij}\, out_i\, out_j - \sum_j I_j\, out_j + \sum_j T_j\, out_j$$

where

E = an artificial n/w energy

w_{ij} = weight from o/p of neuron i to the input of neuron j

Out_j = O/P of neuron j

T_j = external I/p to neuron j

T_j = Threshold of neuron j

The change in energy E due to a change in state of neuron j

$$\delta E = -\left[\sum_{i \neq j} (w_{ij}\, out_i) + I_j - T_j\right] \delta\, out_j$$

$$= -[NET_j - T_j]\, \delta out_j$$

where δout_j is the change in the o/p of neuron j

Suppose that the NET value of neuron j is greater than the threshold. This will cause the term of bracket to be +ve. o/p of neuron j must change in the +ve direction. This means that δout_j can be only +ve or zero. & δE must be -ve, hence n/w energy must either increase or stay constant

Teacher's Signature

Date / /
Page No.

Applications :-

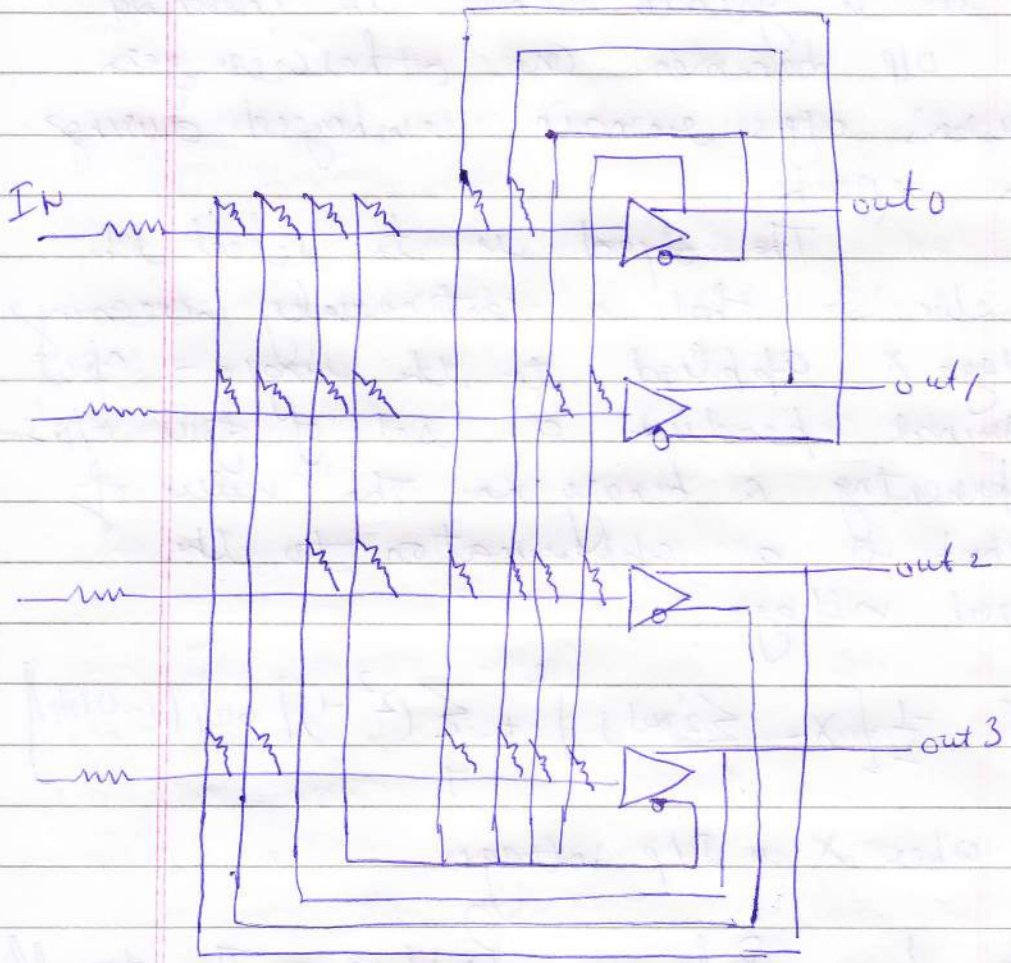
Analog to Digital Converter :- In recent works an electrical signal has been presented that uses a decrement n/w to produce a four bit Analog to digital converter The ckt shows with amplifier serving as artificial neurons Resistor representing weights connect each neuron's o/p to the input of all others To satisfy the stability consistent, no resistor connects a neuron's o/p to its own input and weight are symmetrical i.e resistor from o/p of neuron i to The input of neuron j has the same value as the resistor from the o/p of neuron j to the input of neuron i

Note that the amplifier have both normal & inverting o/p Then takes into account The case in which a weight must be -ve, while permitting the use of ordinary +ve value resistor for all weights However in no case it is necessary to connect both the normal and inverted o/p's of neuron to another neuron input

In realistic ckt, each amplifier will have a finite input

Teacher's Signature

resistance and input capacitance that must be included to characterize the dynamic response n/o. stability does not require that these elements be the same for all amplifier, nor need they be symmetrical



Because these elements affect only the time required to reach a sol^n and not the sol^n itself. They have been omitted to simplify the analysis

The appⁿ assumes that a threshold fⁿ is used. furthermore all of the o/p's are changed at the begining of discrete time intervals called epochs. At the start of each epoch, the summation of the inputs to each neurons is examined If it is greater than the threshold The o/p becomes one, otherwise zero. Neuron's o/ps remain unchanged during an epoch.

The object is to select the resistor, so that a continuously increasing voltage X applied to the single-input terminal produces a set of four o/p's representing a binary no. The value of which is a approximation to the input voltage

$$E = \frac{1}{2}\left[X - \sum_{1} 2^{i} x_i\right]^2 + \sum_{i} (2^{2i}-1)\left[x_i(1-x_i)\right]$$

where X is I/p voltage.

<u>Traveling Salesman Problem</u>:- The travelling salesman problem or TSP is an optimization task that arise in many practical situations. It may be stated as follows given a group of cities to be visited & the

distance b/w each city, find the shortest tour that visit each each city only once and returns to the starting point. The problem has been proven to be one of a large set of problem termed "NP Complete" NP Complete problem have no known method of Sol^n, better than trying all the possibilities. As according to most mathematicians, in any search is generally impractical for more that a few cities, heuristic methods have been applied to find Sol^n That are acceptible is not optimal.

The Sol^n using recurrent n/w's described by Hopfield and Tank is typical in that regard. The result are not guranted to be optimal. Still an answer is reached so rapidly that the technique may proven useful in certain cases.

Suppose that the cities to be visited are lettered A, B, C and D and that the distance b/w the pairs is d_{ab}, d_{bc} and so one. The Sol^n is an ordered set of n cities. The problem is then to map This onto the computational n/w, using neurons in the high gain mode.

Teacher's Signature

Each city is representing as a row of n neurons

One and only one such neuron in a row many be set to one The neuron set to one indicate that the order in which the specific city is visited during the tour fig shows such a result, where city C is visited first. City A is visited second, city D is visited third and city B is visited fourth. The requires n^2 neurons, no that grows rapidly with the no of cities. The length of such a tour would be $d_{ca} + d_{ad} + d_{db} + d_{bc}$ Because each city is visited as at a time. There is only a single 1 in each row and column for an n-city problem there are $n!/2n$ distinct tour. If $n=60$ There are 69.3155×10^{78} possible tours

Considering that there are only 10^{11} stars in milky way galaxy. It becomes clear why calculating all possibilities for 100 city tours would take geological time on the world's fastest computer.

Let us demostrate how to set up a n/w to solve such an NP complete problem Each neuron is identified

by double subscripts indicating the city and the order in which it is visited. For eg out_{xi} indicates that city x was the ith city and the

The energy f^n must satisfy two requirements. First it must be low for only those sol^n that produce a single 1 in column and row second it must favor solutions having short paths.

The first requirement is satisfied by the three-summation energy f^n that follows

$$E = A/2 \sum_x \sum_i \sum_{i \neq j} out_{xi}\, out_{xj}$$

$$+ B/2 \sum_i \sum_x \sum_{y \neq x} out_{xi}\, out_{yi}$$

$$+ C/2 \left[\sum_x \sum_i out_{xi} - n \right]^2$$

where A, B & C are constant. The resulting set of rules as :-

1) The first triple summation is zero if and only if, each row contains no more that single 1
2) The second triple summation is 0 iff each column contains no more single 1

3) The Third requirement is zero off. There are exactly n is in the matrix

City	1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	1	0	0	0
D	0	0	1	0

ORDER VISITED

Traveling Salesman Tour

Back Propagation:- Back propagation is a systematic method for training algorithm multilayer ANN (artificial neural network) It has a mathematical foundation. That is storing if not highly practical. Despite, its limitations back propagation has dramatically expanded. The range of problem to which ANN can be applied & it has generated many successful demonstrations of its power.

The Backpropagation Training Algorithm:-

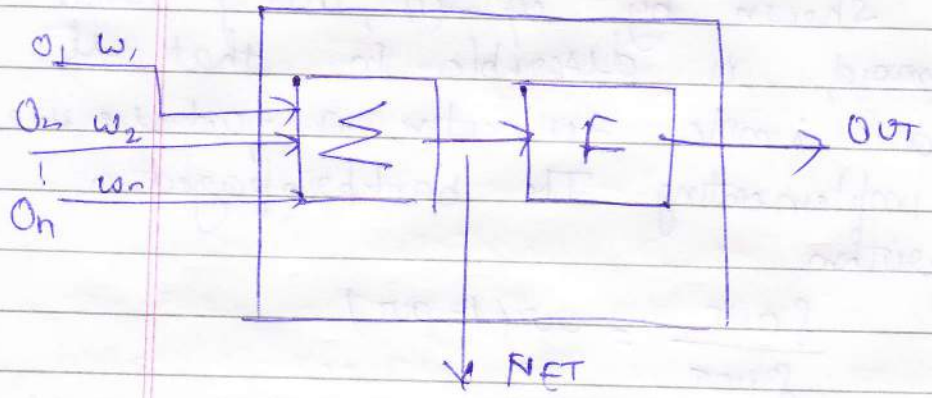
N/W Configuration:- fig shows the neuron used as the foundamental building block for backpropagation n/w's A set of inputs is applied, either

Teacher's Signature

from the outside or from a previous layer. Each of these is multiplied by a weight, and the products are summed. The summation of products is termed NET and must be calculated for each neuron in the n/w. After NET is calculated an activation f^n f is applied to modify it, thereby producing the signal OUT.

Figure shows the activation f^n usually used for backpropagation

$$OUT = 1/(1+e^{-NET})$$



$$NET = O_1\omega_1 + O_2\omega_2 + \dots + O_n\omega_n$$

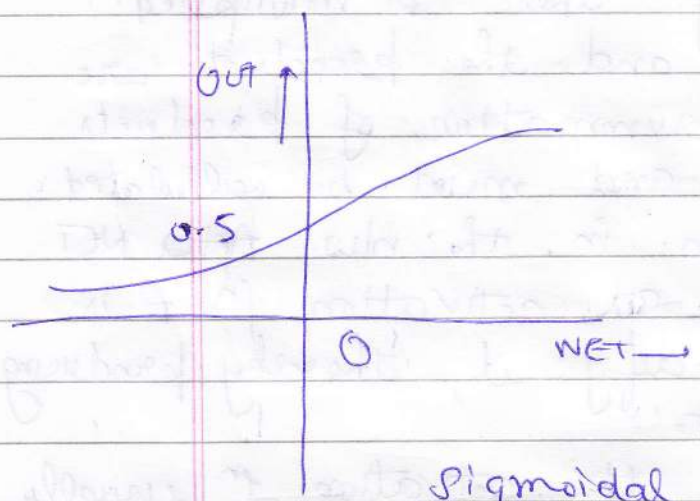
$$= \sum_{i=1}^{n} O_i\omega_i$$

$$OUT = f(NET)$$

Teacher's Signature

Date / /
Page No.

Artificial Neuron with Activation function



Sigmoidal f^n

$$OUT = f(NET) = 1/(1+e^{-NET})$$

$$f'(NET) = \frac{\delta OUT}{\delta NET} = OUT(1-OUT)$$

As shown by eq^n-(1), This f^n called is sigmoid, is discrible in that it has a simple derivative, a fact we use in implementing The backpropagation algorithm

$$\frac{\delta OUT}{\delta NET} = OUT(1-OUT)$$

Sometimes called a logistic, or simply a squashing function The sigmoid clamped compresses the range of NET so that OUT lies b/w zero & one Multilayer n/w's have greater representational power than single layer n/w's only if a nonlinearity is introduced. The squashing f^n produces

Teacher's Signature

the needed nonlinearity

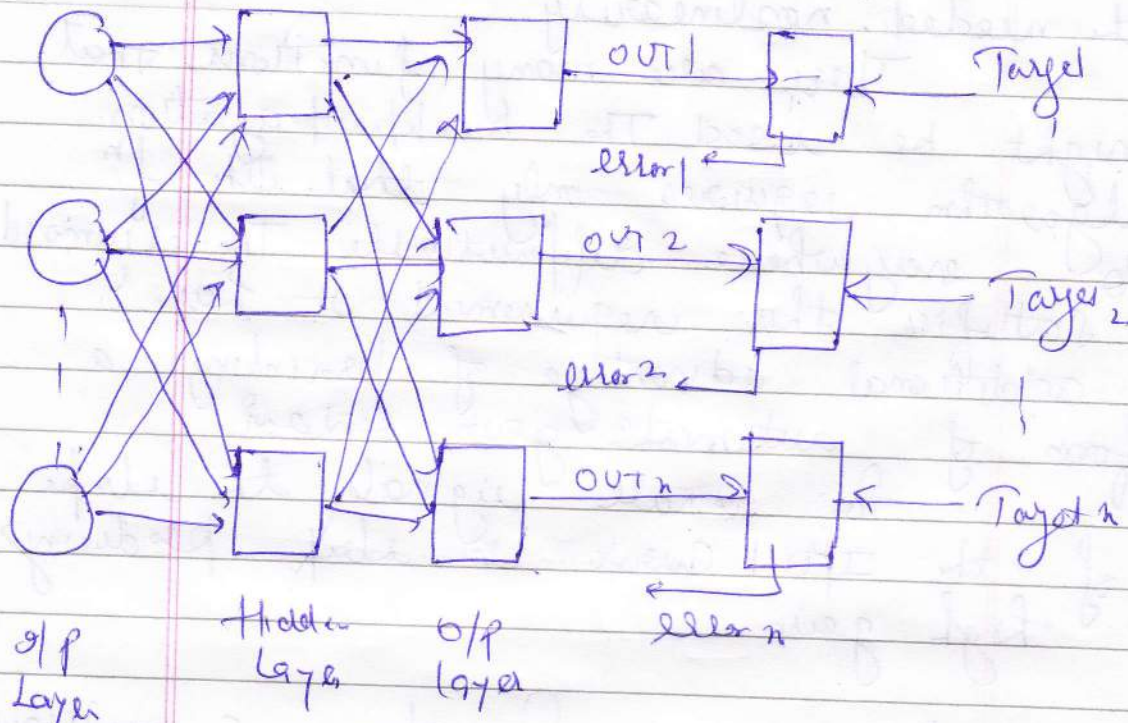
There are many functions that might be used. The backpropagation algorithm requires only that the f^n be everywhere differentiable. The sigmoid satisfies this requirement. It has additional advantage of providing a form of automatic gain control.

for small signals the slope of the I/O curve is steep, producing high gain

Multi layer N/W :- fig shows a multilayer N/W suitable for training with backpropagation. The first set of neurons serve only as distribution points they perform no input summation. The input signal is simply passed through to the weights on their O/P's. Each neuron in the subsequent layer produces NET and OUT signal.

Backpropagation can be applied to N/W with any no of layers. However only two layers of weight with any no of layers are needed to demonstrate the algorithm.

Teacher's Signature



Two Layer backpropagation N/w

<u>An Overview of Training</u> :- The objective of training the n/w is to adjust the weights so that application of set of inputs produces the desired set of o/p's for this, their I/o sets can be reflected as vectors. Training assumes that each o/p vector is paired with a target vector representing the desired o/p. together these are called a training pair.

Usually a n/w is trained over a number of training pairs for example the o/p pair of a pair might consist of a pattern of ones and zeros.

representing a binary image of the alphabet figure shows a set of o/p for letter A drawn on a grid if a line passes through a square. The corresponding neuron input is one, otherwise zero. The output might be a no that represent the letter A or perhaps another set of ones and zeros that could be used to produce an o/p pattern if one wished to train. The n/w to recognize all the letters of the alphabet, 26 training pairs would be required. The group of training pairs is called a training set.

Before starting the training set all of the weights must be initialized to small random numbers. This ensures that the network is not saturated by large values of weights & prevents certain other training pathologies for eg if the weights all start at equal values and the desired performance requires unequal values the n/w will not learn.

Training the backpropagation n/w requires the steps that follow -

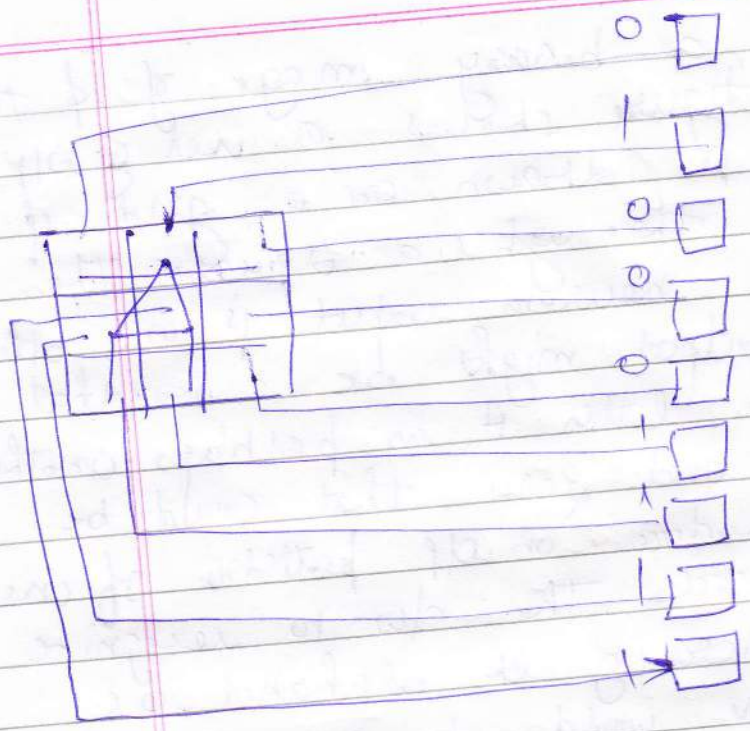


Image Recognition

(1) Select the next training pair from the training set apply the input vector to the n/w input.

(2) Calculate the o/p of the n/w

(3) Calculate the error b/w the n/w o/p & the desired o/p

(4) Adjust the weights of the n/w in a way that minimize the error.

(5) Repeat steps 1 through 4 for each vector in the training set until the error fo

the entire set acceptably low.

The operations required in step 1 and 2 above are similar to the way in which the trained n/w will ultimately be used. that is an input vector is applied and the resulting o/p is calculated. The o/p of the neurons in layer j are calculated then are then used as i/p to layer k. The layer k neurons o/p are calculated & these constitute the n/w o/p vector.

In step 3, each of the n/w o/ps labeled OUT is subtracted from its corresponding components of the target vector to produce an error. The error is used in step 4 to adjust weights of n/w where the polarity and magnitude of the weight changes are determined by the training algorithm.

After enough repetitions of these four steps. The error b/w actual o/ps and target o/ps should be reduced to an acceptable value and the n/w is said to be trained. At this point the n/w is used for recognition and weights are not changed.

It may be seen that step1 and step2 constitute a "forward pass" in that the signal propagates from the n/w input to its o/p.

Step 3 & step4 are reverse pass, here the calculated error signals propagates backward through the n/w where it is used to adjust weights.

<u>Forward Pass</u>:- Step1 & step 2 can be expressed in vector form as follows an input vector X is applied and o/p vector Y is produced. The i/p target vector pair X & T comes from the training set. The calculations is performed on X to produce o/p vector Y.

As we have seen, calculations in multilayer n/w is done layer by layer starting at the layer nearest to the inputs. The net value of each neuron in first layer is calculated as the weight sum of its neuron inputs. The activation fn f then "squashes" the set of o/p for a layer is found. It serves as input to the next layer. The process is repeated layer by layer, until the final set of

n/w o/p is produced.

This process can be stated succinctly in vector notation. The weights b/w neurons can be considered to be a matrix W. For eg the weight from neuron 8 in layer 2 to be matrix from neuron 5 in layer 3 is designated $W_{8,5}$ Rather than using the summation of products the net vector for a layer N may be expressed as the product of X and W in vector notation $N = XW$ Applying the f^n f to the NET vector N component by component produces the o/p vector O. Thus for a given layer the following expression describes the calculation process.

$$O = f(XW) \quad \text{—} \quad (1)$$

The o/p vector for one layer is the input vector for the next so calculating the o/p's of the final layer requires the applications of eq (1) to each layer from the n/w I/P to its o/p.

Teacher's Signature

Date / /
Page No.

Reverse Pass:-

Adjusting the weights of O/P Layer:- Because the target value is available for each neuron in the O/P layer, adjusting the associated weights easily accomplished using a modification of delta rule. Interior layers are referred to as "hidden layer" as their O/P have no target values for comparision. Hence training is more complicated. fig shows the training process for a single weight from neuron p in hidden layer j to neuron q in O/P layer k.

The O/P of layer k subtracted from its target value to produce an error signal. This is multiplied by the derivative of the sigmoid fn $[OUT(1-OUT)]$ calculated for that layer's neuron k, thereby producing the δ values

$$\delta = OUT(1-OUT)(Target - OUT) \quad \text{--- (1)}$$

Then δ is multiplied by OUT_j from a neuron j, the source neuron for the weight in questions. This product is turn multiplied by a training rate coefficient from 0.01 to 1.0 & the result

Teacher's Signature

is added to the weight. An identical process is performed for each weight proceeding from a neuron in the hidden layer to a neuron in o/p layer.

The following eqn illustrate this calculations

$$\Delta W_{pq,k} = \eta \, \delta_{q,k} \, OUT_{pj} \quad — (2)$$

$$W_{pq,k}(n+1) = W_{pq,k}(n) + \Delta W_{pq,k} \quad — (3)$$

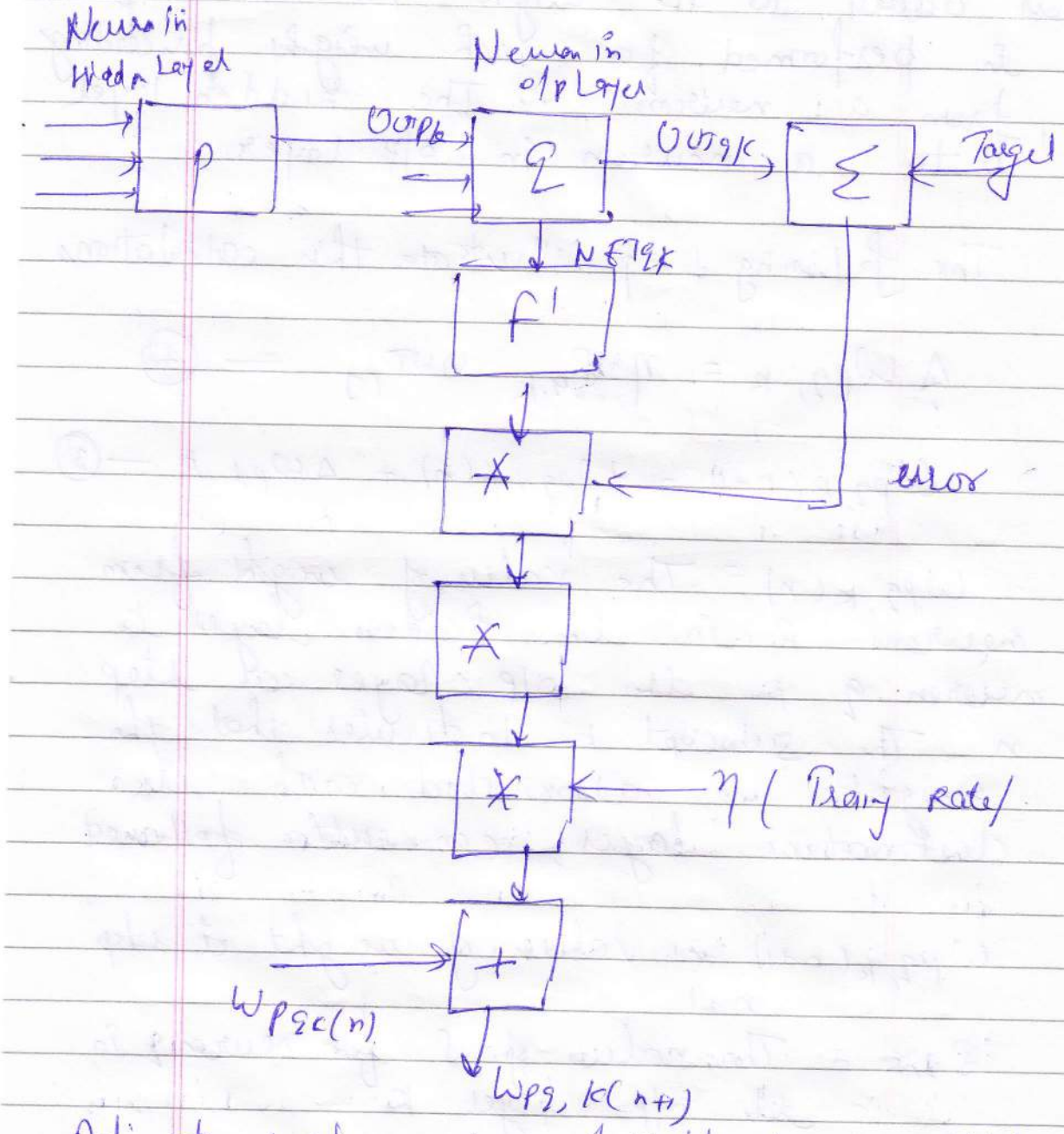
$W_{pq,k}(n)$ = The value of weight from neurons p in the hidden layer to neuron q in the o/p layer at step n. The subscript k indicates that the weight is associated with the destination layer, convention followed

$W_{pq,k}(n+1)$:- value of weight at step n+1

$\delta_{q,k}$ = The value of δ for neuron q in the o/p layer k

$OUT_{p,j}$ = The value of OUT for neuron p in the hidden layer j

Teacher's Signature



Adjusting the weight of Hidden Layer :- Hidden Layer have no target vector, so that training process described above can not be used. The lack of a training target signed efforts to train multilayer N/w until

Date / /
Page No.

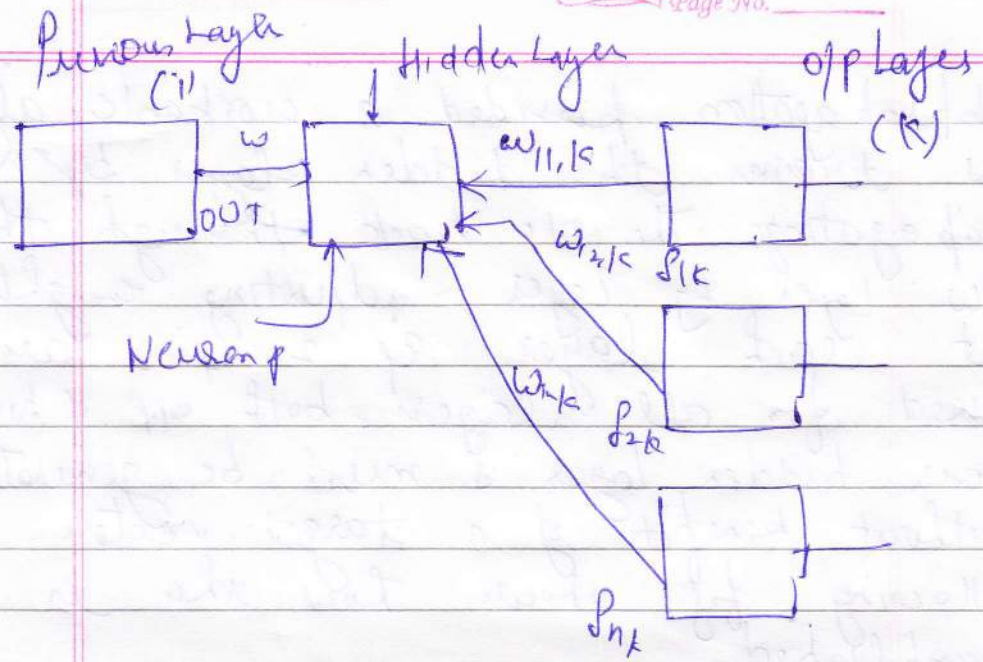
backpropagation provided a workable alg. BPN trains the hidden layer by propagating the o/p back through the n/w layer by layer, adjusting weights at each layer. Eq 2 & 3 are used for all layers, both o/p & hidden. However hidden layer δ must be generated without benefit of a target vector. Following fig shows how this is accomplished.

→ first δ is calculated for each neuron in the o/p layer as in eq (1). It is used to adjust the weight feeding into the o/p layer.

→ Then it is propagated back through the same weight to generate a value for δ for each neuron in the first hidden layer.

→ These values of δ are used, in turn adjust the weight of the hidden layer and in similar way are propagated back to all preceding layers.

Teacher's Signature



Training a weight in Hidden Layer

Consider a single neuron in Hidden Layer just before the o/p Layer. In the forward pass The neuron propagates its o/p value to neuron in the o/p Layer through the interconnecting weight. During training these weight operate in reverse, passing the value of δ from the o/p Layer back to the hidden Layer. Each of these weight is multiplied by the δ value of the neuron to which it connects in o/p Layer. The value of δ needed for the hidden Layer neurons is produced by summing all such products and multiply by the derivative of the

Squashing f^n

$$\delta_{pj} = out_{pj}(1 - out_{pj}) \sum_q \delta_{qk} \cdot w_{pqk})$$

with δ in hand. The weights feeding the first hidden layer can be adjusted using eq^n (2) & (3) modifying indices to indicate the correct layer.

For each neuron in given layer i.e hidden layer. δ_k must be calculated & all the weights associated with that layer must be adjusted. This is repeated moving back towards the input layer by layer until all weights are adjusted.

<u>Applications</u>:- Back propagation has been applied to a wide variety of research applications A NEC in Japan has associated recently that is has applied back by improving accurately to over 99% to a new optical character recognition system. This achievement can be through combinational of providing conventional n/w algorithms.

Teacher's Signature

Date / /
Page No.

Local minima:- BPN employs a type of gradient descent, that is follows the slope of error surface downward, constantly adjusting the weight toward a minimum. The error surface of a complex n/w is highly connected. The n/w can get trapped in a local minimum, where there is a much deeper minimum nearby. From the limited viewpoint of the n/w all directions are up and it has no way to escape statistically training method can help avoid this trap, but tends to be slow. Wasserman has proposed a method that combines the statistically methods of Cauchy m/c with the gradient descent of BPN to produce a system. That finds global minima while retaining the higher training rate of BPN.

Step Size:- A careful reading of the convergence proof of Rumelhart, Hinton shows that infinitesimally small weight adjustments are assumed. This is clearly impractical as it implies infinite training time. It is necessary to select a finite step size. There is very little to guide that decision other than

Teacher's Signature

Counter propagation Network:- CPN developed by Robert Hecht Nielson is beyond the representation of limit of single layer network. This is a multilayer Network based on various combining structure of i/p, clustering and output layers It deduced the time by one hundred times. CPN is different than backpropagation Network sense that it provides solutions for those application which can not have larger iterations CPN can be used for data compression, approximation of f^n, pattern association and signal enhancement applications.

It is the combination of two algorithm : self organizing map of Kohonen & Grossberg out star.

The CPN as a look up table capable of generalization. The training process involve i/p vector with corresponding o/p vector

CPN is trained in two stages

stage 1:- The i/p vector are clustered on the basis of Euclidean distance or by the dot product method.

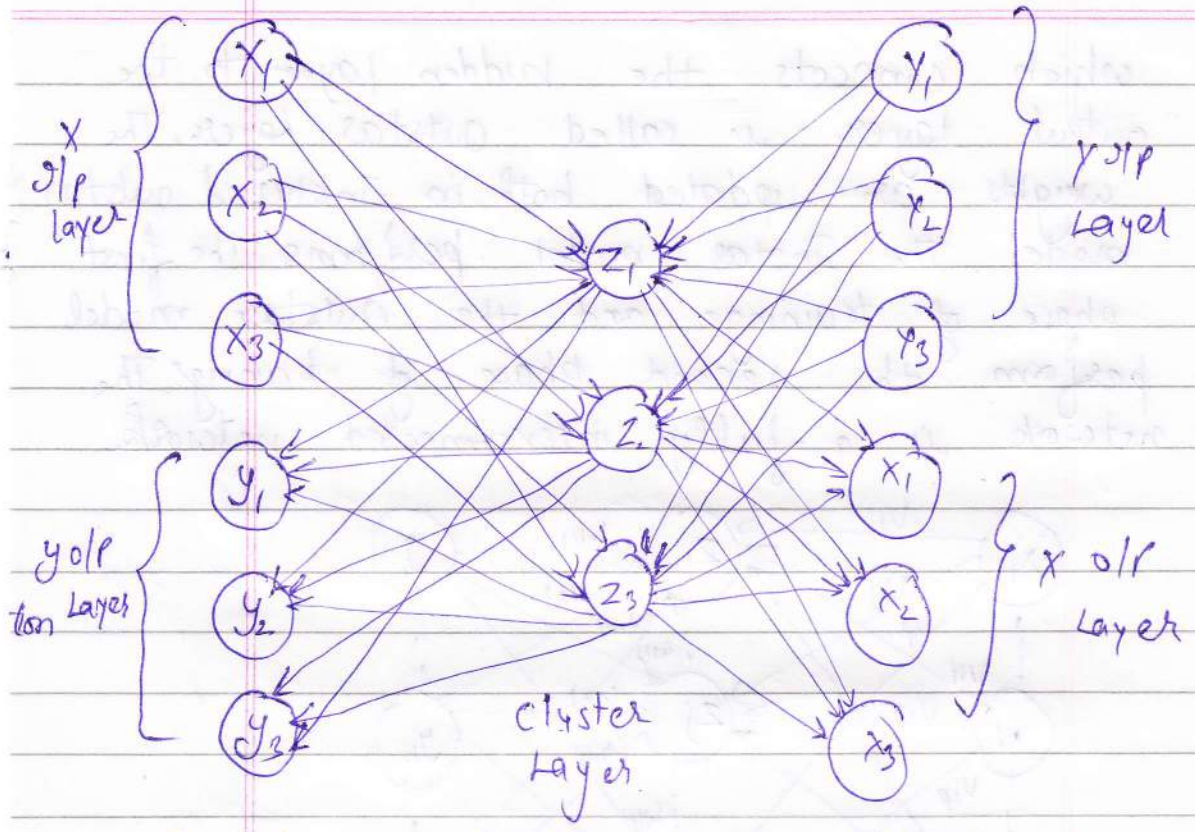
stage 2:- The desired response is obtained by adopting the weight from the cluster units to the o/p units

CPN is classified in two types :-
(1) Full counter propagation Network
(2) Forward only counter propagation Network
The CPN trains rapidly. If appropriately applied it can save large amount of computing time. It is also useful for rapid prototyping of systems.

Full Counter Propagation Network :- The full CPN possess the generalization capability which allows it to produce a correct o/p even when it is given an i/p vector that is partially incomplete or partially incorrect. full CPN can represent large no of vectors pairs x:y by constructing a look up table.
During the first stage, the training vector pairs are used to form clusters. clustering can be done either by dot product or by euclidean distance. During second phase the weight are adjusted b/w the cluster units and the output units.

Architecture :- The given vector pair are x:y and the approximated vector pair may be x*:y*

Teacher's Signature

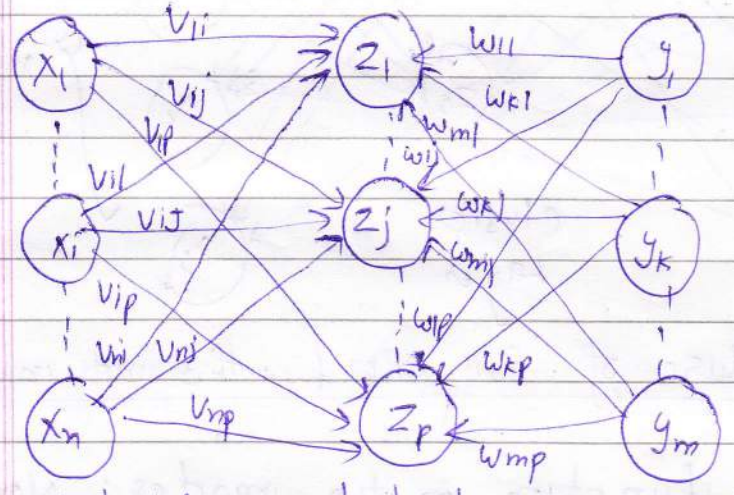


Architecture of full CPN (with 3 units in all layers)

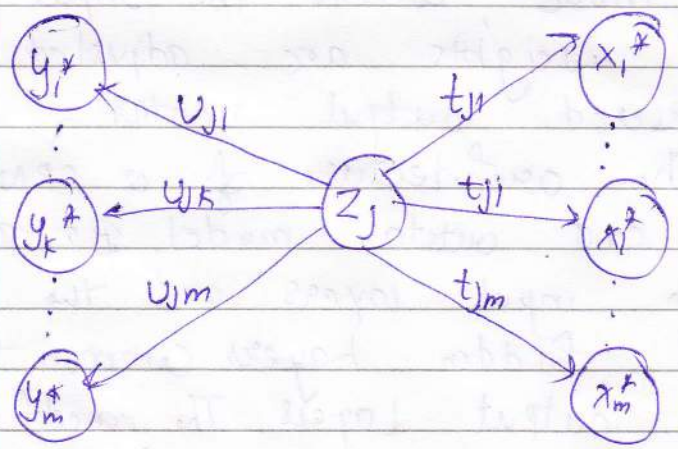
The CPN function in two modes: Normal mode, where input vector is accepted and output vector is produced and training mode where the input vector and the weights are adjusted to obtain desired output vector.

The architecture of a CPN resembles an instar and outstar model. Basically, it has two input layers and two output layers with hidden layers common to the input and output layers. The model which connects the i/p layer to the hidden layer is called Instar model and the model

which connects the hidden layer to the output layer is called Outstar Layer. The weights are updated both in Instar & outstar model. The Instar model performs the first phase of training and the Outstar model perform the second phase of training. The network is a fully interconnected network.



first phase of full CPN



y output x output

Second phase of full CPN

Date / /
Page No.

Training phases of FULL CPN:-

First phase:- This phase of training may be called as Instar modeled training. The active units here are the units in the x-input, z-cluster, and y-input layers.

$$V_{ij}(new) = V_{ij}(old) + \alpha(x_i - V_{ij}(old))$$
$$(1-\alpha)V_{ij}(old) + \alpha x_i, \quad i=1 \text{ to } n$$
$$W_{kj}(new) = W_{kj}(old) + \beta(y_k - W_{jk}(old))$$
$$(1-\beta)W_{kj}(old) + \beta y_k, \quad k=1 \text{ to } m$$

Second phase:- In this phase we can find only the J unit remaining active in the cluster layer. The weights from the winning cluster unit J to the output units are adjusted so that vector of activation of units in the y output layer y^*, is app. of i/p vector y; & x^* is an approximation of i/p vector x. This phase may be called the outstar modeled training. The weight updation is done by the Grossberg learning rule, which is used only for outstar learning. In outstar learning, the weight updation rule is given by.

$$u_{jk}(new) = u_{jk}(old) + \alpha(y_k - u_{jk}(old))$$
$$(1-\alpha)u_{jk}(old) + ay_k, \quad k=1 \text{ to } m$$
$$t_{ji}(new) = t_{ji}(old) + b(x_i - t_{ji}(old))$$
$$(1-b)t_{ji}(old) + bx_i, \quad i=1 \text{ to } n$$

Date / /
Page No.

Training Algorithm:-

The parameters used are :-

x = Input training vector $x = (x_1 \ldots x_i \ldots x_n)$

y = target o/p vector $y = (y_1 \ldots y_k \ldots y_m)$

Z_j = activation of cluster unit Z_j

x^* - approximation to vector x

y^* - approximation to vector y

v_{ij} - weight from x o/p Layer to Z cluster

w_{jk} - weight from y o/p Layer to Z cluster

t_{ji} - weight from cluster layer to X o/p layer

u_{jk} - weight from cluster layer to Y o/p layer

α, β - learning rates during kohonen learning

a, b - learning rates during Grossberg learning

Step-1: Initialize the weights, learning rates

Step-2 while stopping condition for phase 1 training is false, perform step 3-8

Step-3: For each training input pair do step 4-6

Step-4: Set x input layer activations to vector x, set x input layer activations to vector y

Step 5: find winning cluster unit using Euclidean distance

Step 6: Update weight for winning unit

$$v_{ij}(new) = v_{ij}(old) + \alpha(x_i - v_{ij}(old)) \; ; \; i = 1 \text{ to } n$$

$$w_{kj}(new) = w_{kj}(old) + \beta(y_k - w_{kj}(old)) \; ; \; k = 1 \text{ to } m$$

Teacher's Signature

Date / /
Page No.

Step 7: Reduce learning rates α and β

Step 8: Two stopping conditions for phase 1 training

Step 9: While stopping condition for phase 2 training is false, perform step 10-16

Step 10: For each training input pair x:y do step 11 - 14

Step 11: Set X i/p layer activations to vector x;
Set Y i/p layer activations to vector y

Step 12: find winning cluster unit using Euclidean distance

Step 13: Updating the weights for winning unit, the value of α & β in this phase are constant

$v_{ij}(new) = v_{ij}(old) + \alpha(x_i - v_{ij}(old))$; $i = 1$ to n

$w_{kj}(new) = w_{kj}(old) + \beta(y_k - w_{kj}(old))$, $k = 1$ to m

Step 14: Update weights from unit Z_j to the o/p layer

$U_{jk}(new) = U_{jk}(old) + a(y_k - U_{jk}(old))$; $k = 1$ to m

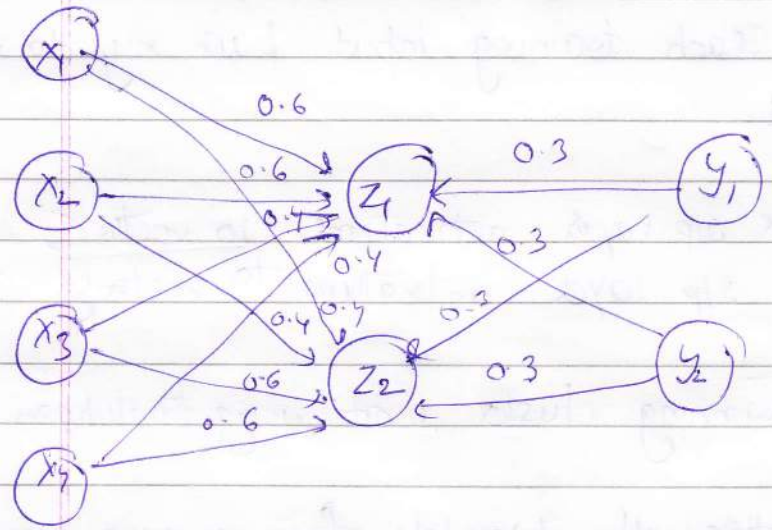
$t_{ji}(new) = t_{ji}(old) + b(x_i - t_{ji}(old))$; $i = 1$ to n

Step 15: Learning rates a & b are to be reduced

Step 16: Test the stopping conditions for phase 2 training.

Teacher's Signature

Example 10.1 :- Consider the following full counter propagation. Using the i/p pair $x = (1\ 0\ 1\ 0)$, $y = (1\ 0)$, Perform the first phase of training. Find the activation of the cluster layer units. Update the weight using a learning rate of 0.2.



Solution :-

1) Initialize weights

$$V = \begin{bmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix}, \quad W = \begin{bmatrix} 0.3 & 0.3 \\ 0.3 & 0.3 \end{bmatrix}$$

$$\alpha = 0.2 \quad \& \quad \beta = 0.2$$

2) Begin training

Step 7: Reduce learning rates α and β

Step 8: Two stopping conditions for phase 1 training

Step 9: While stopping condition for phase 2 training is false, perform step 10-16

Step 10: For each training input pair x:y do step 11 - 14

Step 11: Set X i/p layer activations to vector x;
Set Y i/p layer activations to vector y

Step 12: find winning cluster unit using Euclidean distance

Step 13: Updating the weights for winning unit, the value of α & β in this phase are constant

$$v_{ij}(new) = v_{ij}(old) + \alpha(x_i - v_{ij}(old)); \quad i = 1 \text{ to } n$$

$$w_{kj}(new) = w_{kj}(old) + \beta(y_k - w_{kj}(old)); \quad k = 1 \text{ to } m$$

Step 14: Update weights from unit Z_j to the o/p layer

$$U_{jk}(new) = U_{jk}(old) + a(y_k - U_{jk}(old)); \quad k = 1 \text{ to } m$$

$$t_{ji}(new) = t_{ji}(old) + b(x_i - t_{ji}(old)); \quad i = 1 \text{ to } n$$

Step 15: Learning rates a & b are to be reduced

Step 16: Test the stopping conditions for phase 2 training.

Teacher's Signature

Date / /
Page No.

3) Present the i/p vector pair

$$X = (1\,0\,1\,0)$$
$$Y = (1\,0)$$

4) Setting the activation to X & Y

5) $D(j) = \sum_i (x_i - v_{ij})^2 + \sum_k (x_k - w_{jk})^2$

$D(1) = (1-0.6)^2 + (0-0.6)^2 + (1-0.4)^2 + (1-0.3)^2 + (0-0.3)^2 = 1.62$

$D(2) = (1-0.4)^2 + (0-0.4)^2 + (1-0.6)^2 + (0-0.6)^2 + (1-0.3)^2 + (0-0.3)^2 = 1.62$

$D(1) = D(2)$ Therefor $J=1$

6) Weight updation and $J=1$

$V_{ij}(n) = V_{ij}(0) + \alpha(X_i - V_{ij}(0))$; $i=1 \to 4$

$V_{11}(n) = 0.6 + 0.2(1-0.6) = 0.68$

$V_{21}(n) = 0.6 + 0.2(0-0.6) = 0.48$

$V_{31}(n) = 0.4 + 0.2(1-0.4) = 0.52$

$V_{41}(n) = 0.4 + 0.2(0-0.4) = 0.32$

Also

$W_{kj}(new) = W_{kj}(old) + \beta(y_k - W_{kj}(old))$; $k = 1 \to 2$

$W_{11}(n) = W_{11}(0) + \beta(y_1 - W_{11}(0))$

$W_{11}(n) = 0.3 + 0.2(1-0.3) = 0.44$

$W_{21}(n) = 0.3 + 0.2(0-0.3) = 0.24$

The updated weight are

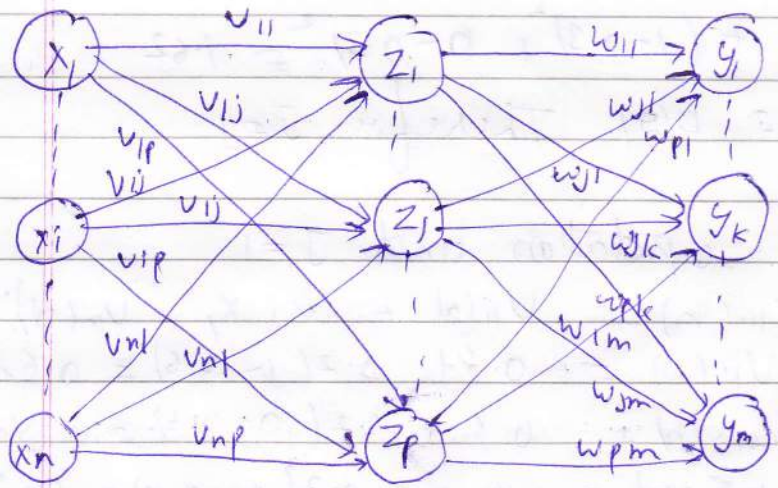
$$V = \begin{bmatrix} 0.68 & 0.4 \\ 0.48 & 0.4 \\ 0.52 & 0.6 \\ 0.32 & 0.6 \end{bmatrix} \quad W = \begin{bmatrix} 0.44 & 0.3 \\ 0.24 & 0.3 \end{bmatrix}$$

Teacher's Signature

forward only Counter Propagation Network:-

forward only is a simplified form of full CPN. This net may be used if the mapping from x to y is well defined, but the vice versa is not so. It is different from the full CPN in the sense that it uses only x-vectors to form the clusters on the kohonen units during the first stage of training.

Architecture:-



Architecture of forward only CPN

Training Algorithm:-

0) Initialize weights, learning rates
1) for each training i/p x perform step 3-5
2) while stopping conditions for phase 1 is false, perform step 2-7

Teacher's Signature

3) Set x i/p Layer activations to vector x
4) Find winning cluster unit
5) Update weights on winning cluster unit

$$V_{ij}(new) = V_{ij}(old) + \alpha(x_i - V_{ij}(old))$$
$$i = 1 \text{ to } n$$

6) Reduce learning rate α
7) Test stopping condition for Phase 1 training
8) While stopping condition is false for phase 2 training do steps 9-15
9) for each training i/p pair x, y do step 10-13
10) Set x i/p layer activations to vector x
Set y o/p layer activations to vector y
11) complete the winner cluster unit
12) Update weight in to unit J

$$V_{ij}(new) = V_{ij}(old) + \alpha(x_i - V_{ij}(old))$$
$$i = 1 \text{ to } n$$

13) update weight from cluster unit

$$w_{jk}(new) = w_{jk}(old) + a(y_k - w_{jk}(old))$$
$$; k = 1 \text{ to } m$$

14) Reduce learning rate a
15) Test stopping condition for phase 2

$\alpha = 0.5$ to 0.8

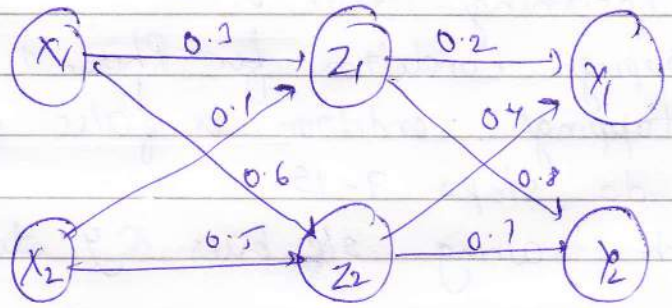
$a = 0$ to 1

In Euclidean distance

$$D(J) = \sum_i (x_i v_{ij})^2$$

Teacher's Signature

Q:- Consider the forward only CPN shown in fig below using i/p pair $X = (1, 0)$, $y = (0, 1)$ perform the training (one step - one iteration). find the activation of cluster layer unit Update weight using learning $\alpha = 0.5$ & $a = 0.1$



Sol^n 0) $v = \begin{bmatrix} 0.3 & 0.6 \\ 0.1 & 0.5 \end{bmatrix}$ $w = \begin{bmatrix} 0.2 & 0.8 \\ 0.4 & 0.7 \end{bmatrix}$

$\alpha = 0.5, \quad a = 0.4$

1) Begin training of first phase

2) Present i/p vector $X = (1\ 0)$

3) Set activations of $X = (1\ 0)$

4) Calculate winning cluster

$$D(j) = \sum_i (x_i - v_{ij})^2, \quad i = 1 \text{ to } 2$$

$$D(1) = \sum_{i=1}^{2} (x_i - v_{i1})^2$$

$$D(1) = (1 - 0.3)^2 + (0 - 0.1)^2 = 0.5$$

$$D(2) = (1 - 0.6)^2 + (0 - 0.5)^2 = 0.41$$

$$D(2) < D(1)$$

$$J = 2$$

Teacher's Signature

Date / /
Page No.

5) Updating the weights on the winner unit

$$V_{in(new)} = V_{ij}(old) + \alpha(X_i - V_{ij}(old))$$

$i = 1$ to 2

$$V_{12}(n) = V_{ij}(old) + \alpha(X_i - V_{ij}(old)), i=$$

$$V_{12}(n) = 0.6 + 0.5(1-0.6) = 0.8$$

$$V_{22}(n) = 0.5 + 0.5(0-0.5) = 0.25$$

$$\begin{bmatrix} 0.3 & 0.8 \\ 0.1 & 0.25 \end{bmatrix}$$

6-7 Keep α some constant value for using it in the second phase of stopping conditions as mentioned check or move to next step

8) Begin second phase of training

9) Present i/p vector pair x & y

10) $X = (1\ 0)$ $\quad y\ (0\ 1)$

11) $D(i) = \sum_i (x_i - V_{ij})^2$

$$D(1) = (1-0.3)^2 + (0-0.1)^2 = 0.5$$

$$D(2) = (1-0.8)^2 + (0-0.25)^2 = 0.1025$$

$$D(2) < D(1)$$

$$J = 2$$

12) $V_{12}(n) = V_{12}(o) + \alpha(x_i - V_{12}(old))$

$$V_{12}(n) = 0.8 + 0.5(1-0.8) = 0.9$$

$$V_{22}(n) = 0.25 + 0.5(0-0.25) = 0.125$$

13) $W_{21}(n) = W_{21}(0) + 0.1(y_1 - W_{21}(0))$

$$W_{21}(n) = 0.4 + 0.1(0-0.4) = 0.36$$

$$W_{22}(n) = 0.7 + 0.1(1-0.7) = 0.73$$

Teacher's Signature

updated weight of second phase of training

$$V = \begin{bmatrix} 0.3 & 0.9 \\ 0.1 & 0.125 \end{bmatrix} \quad W = \begin{bmatrix} 0.2 & 0.5 \\ 0.36 & 0.73 \end{bmatrix}$$

Application of CPN -

(a) Data Compression :- (1) CPN can be used to compress data to transmission reducing the number of bits to be sent

(2) Suppose an image is to be transmitted

(3) It can be divided into subimages

(4) Each sub images is further divided into pixels

(5) Each subimage is now a vector of pixels

(6) If there are n pixels in a sub image, then n bits will be required to transmit it

(7) CPN can be used to perform vector quantization

(8) The set of sub image vectors is used as input to train the kohonen layer in accretive mode in which only one neuron is activated

(b) Clustering :- Clustering is basically the grouping of objects according to their similarity. We can see how kohonen network can perform clustering

through competitive learning called winner take all.

The node with the largest activation level is declared the winner the competition. This node is the only node will generate the output signal, and all other nodes are suppressed to the zero activation level.

Kohonen network uses intra layer connections to moderate this competition. The output of each node act as an inhibitory i/p to the other nodes but is actually excitatory in its neighbourhood.

After training the weight vector of each node encodes the information of a group of similar i/p pattern. Given an i/p vector it is assigned to the node with the maximum activation. Since the number of nodes are fixed. This kind of algorithm is more noise tolerant than algorithm such as ART networks.

(c) <u>Statistical Modeling</u>:- The Kohonen network has the ability to extract the statistical properties of the i/p data set. It can specifically estimate the probability density f_n of i/p data. This property is very useful since we can apply Kohonen network to statistical modelling. It is a fully trained

network.

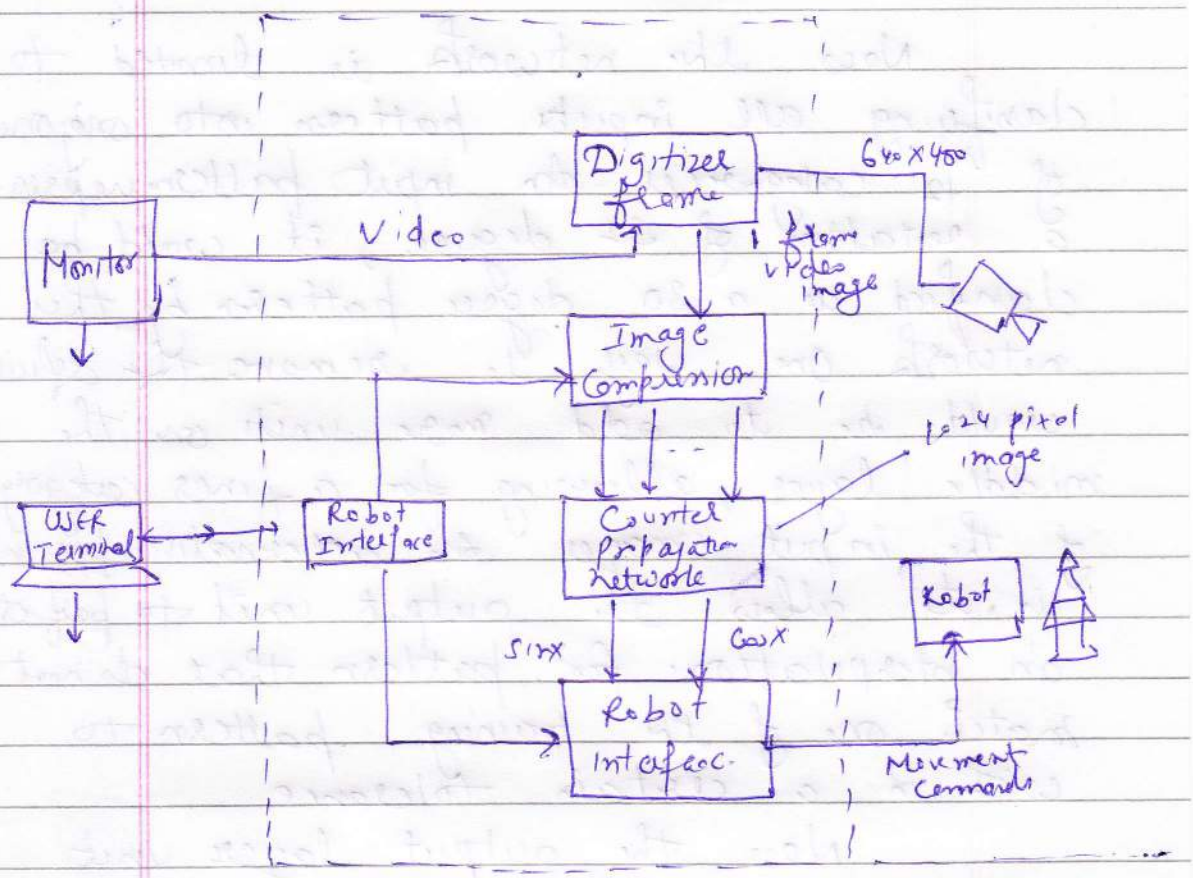
Under the winner take all strategy only one neuron gets activated for each i/p pattern. This is called accretive mode. Another mode called the interpolative mode permit a group of neurons having the highest activations to be winner at one time. This allows the output of the network to be an interpolation of more than one best answer and is thus capable of representing more complex pattern and producing more accurate results.

Image classification using CPN:-

Here we look how CPN can be used to classify images into categories.

The problem is to determine the angle of rotation of the principle axis of an object in two dimensions, directly from the raw video image of the object. In this case the object is the model of the space shuttle which can be rotated 360° about a single axis of rotation. Numerical algorithm as well as pattern matching techniques exist that will solve this problem. The neural network solutions poses some

interesting examples.



The system uses a CPN having 1026 input units, 12 hidden units & 2 output units. The unit on the middle layer learn to divide the input vectors into different classes. There are 12 units in this layer and 12 different input vector are used to train the network. These 12 vector represents images of the shuttle at 30 degree increments (0°, 30°, ... 330°) since there are 12 categories and 12 training vectors, training of competitive layer consist of setting each

unit weight equal to one of the input vector

Now this network is limited to classifying all inputs pattern into only one of 12 categories. An input pattern represents a rotation of 32 degree, it would be classified as a 30 degree pattern by this network. One way to remove this deficiency would be to add more units on the middle layer, allowing for a finer category of the input images. An alternative approach is to allow the output unit to perform an interpolation for pattern that do not match one of the training pattern to with in a certain tolerence.

Now the output layer unit calculate their output values according to eqn $Y'_k = \sum w_{kj} Z_j$. In the normal case where the ith hidden unit wins $Y'_k = w_{ki}$ since $Z_j = 1$ for $J = 1$ & $Z_j = 0$ otherwise should two competitive units shared in winning the one with the two closet matching input pattern is that $Z_j \times \cos\theta_j$ for the two winning units. If we restrict the total output from the middle layer of unity then the output values from the output layer would be -

Teacher's Signature

$$y'_k = w_{ki} z_i + w_{kj} z_j$$

where the ith & jth units on the middle layer were the winner

$$z_i + z_j = 1$$

The network output is a linear interpolation of the output that would be obtained from the two pattern that exactly matched the two hidden the units that shared the victory.

One of the benefit of using the neural network approach to pattern matching is robustness in the presence of noise or of contradictory data.

Teacher's Signature